

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
28 June 2007 (28.06.2007)

(10) International Publication Number
WO 2007/071116 A1

(51) International Patent Classification:
G06F 15/16 (2006.01)

Xin [CN/CN]; Room 302, N0. 264 South Shanxi Road, Shanghai 200001 (CN).

(21) International Application Number:
PCT/CN2005/002304

(74) Agent: CHINA PATENT AGENT (H.K.) LTD.; 22/F, Great Eagle Centre, 23 Harbour Road, Wanchai, Hong Kong (HK).

(22) International Filing Date:
23 December 2005 (23.12.2005)

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(25) Filing Language:
English

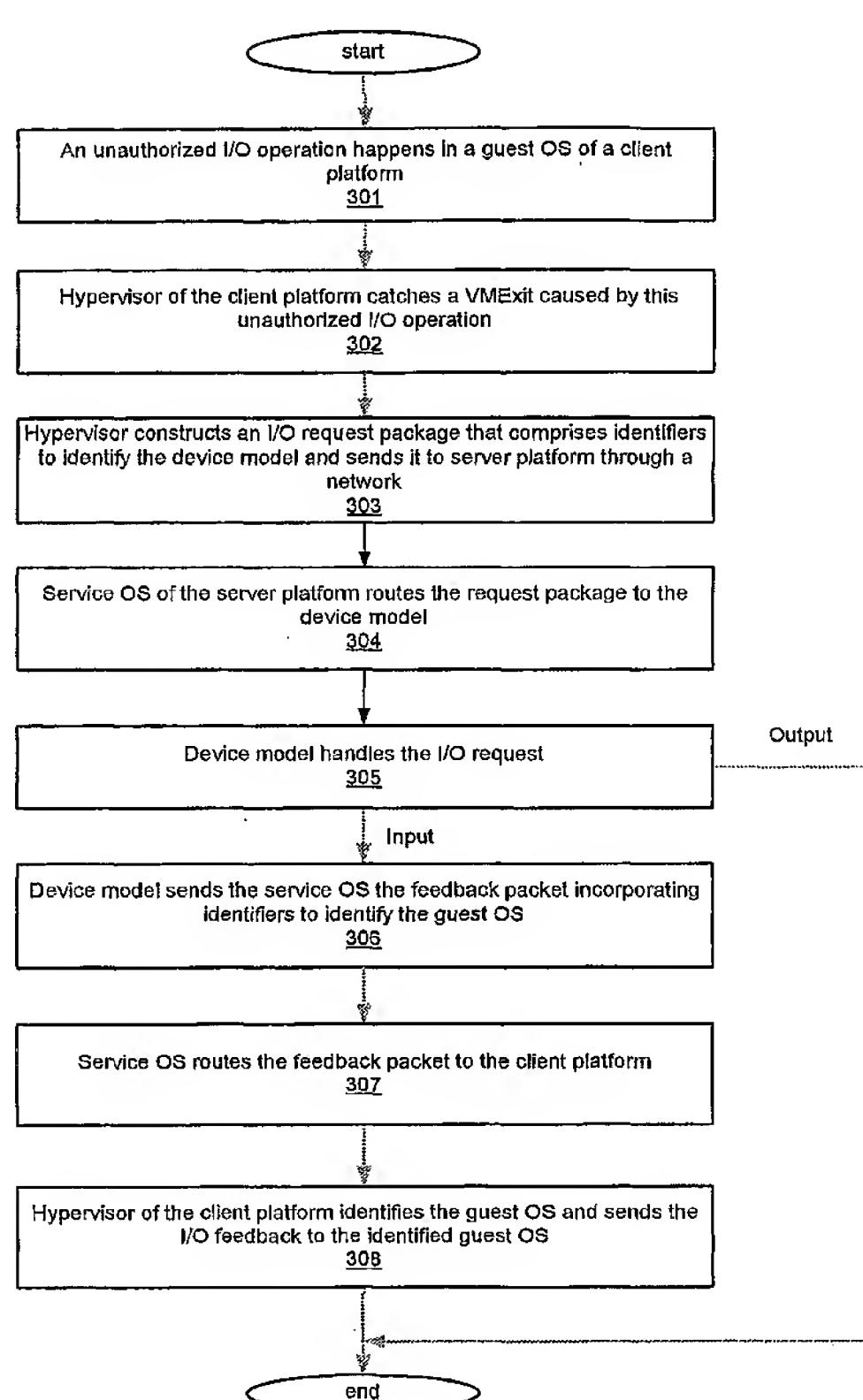
(26) Publication Language:
English

(71) Applicant (for all designated States except US): INTEL CORPORATION [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT,

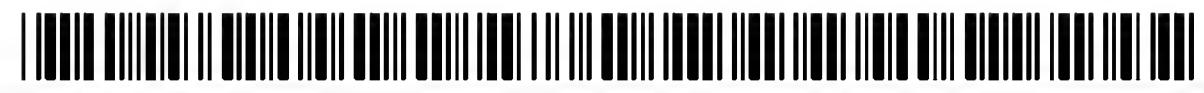
[Continued on next page]

(54) Title: MANAGING DEVICE MODELS IN A VIRTUAL MACHINE CLUSTER ENVIRONMENT



(57) Abstract: Machine-readable media, methods, apparatus and system are described. In some embodiments, a client platform may determine that an input/output operation related to a hardware device may happen in a virtual machine of the client platform and request a server platform via the network to handle the input/output operation related to the hardware. The server platform may receive the request from the client platform through the network; and identify a software device from a plurality of software devices in the server platform to handle the request, wherein the identified software device corresponds to the hardware device related to the input/output operation.

WO 2007/071116 A1



RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Published:

— *with international search report*

MANAGING DEVICE MODELS IN A VIRTUAL MACHINE CLUSTER ENVIRONMENT

BACKGROUND

[0001] A virtual machine (VM) architecture logically partitions a physical machine, 5 such that the underlying hardware of the machine is time-shared and appears as one or more independently operation virtual machines. A computer platform in a virtual machine cluster environment may comprise a virtual machine monitor (VMM) that may create a plurality of virtual machines and runs on the computer platform to facilitate for other software the abstraction of one or more 10 virtual machines. The computer platform may further comprise a plurality of device models that may be virtualizations/simulations of real devices. The virtual machine monitor may manage resource usage for operations between the virtual machines and the device models, such as device related input/output operation and device model initiated interrupt. Examples for the 15 virtual machine monitor may comprise a hybrid virtual machine monitor, a host virtual machine monitor and a hypervisor virtual machine monitor. Examples for the real device may comprise various input/output (I/O) devices, an interrupt controller, and an event timer, etc.

BRIEF DESCRIPTION OF THE DRAWINGS

20[0002] The invention described herein is illustrated by way of example and not by way of limitation in the accompanying figures. For simplicity and clarity of illustration, elements illustrated in the figures are not necessarily drawn to scale. For example, the dimensions of some elements may be exaggerated relative to other elements for clarity. Further, where considered appropriate,

reference labels have been repeated among the figures to indicate corresponding or analogous elements.

[0003] Fig. 1 shows an embodiment of a virtual machine cluster environment.

[0004] Fig. 2 shows an embodiment of structures of a server platform and a plurality of client platforms in the virtual machine cluster environment of Fig. 1.

5 [0005] Fig. 3 shows an embodiment of a method of implementing a device related input/output operation in the virtual machine cluster environment of Fig. 2.

[0006] Fig. 4 shows an embodiment of a method of implementing a device model initiated interrupt in the virtual machine cluster environment of Fig. 2.

10[0007] Fig. 5 shows an embodiment of a computer platform that may be implemented as a platform within the virtual machine cluster environment of Fig. 2.

[0008] Fig. 6 shows another embodiment of structures of a server platform and a plurality of client platforms in the virtual machine cluster environment of Fig. 1.

15 DETAILED DESCRIPTION

[0009] The following description describes techniques for managing device models in a virtual machine cluster environment. In the following description, numerous specific details such as logic implementations, pseudo-code, means to specify operands, resource partitioning/sharing/duplication implementations, 20 types and interrelationships of system components, and logic partitioning/integration choices are set forth in order to provide a more thorough understanding of the current invention. However, the invention may be practiced without such specific details. In other instances, control structures, gate level circuits and full software instruction sequences have not

been shown in detail in order not to obscure the invention. Those of ordinary skill in the art, with the included descriptions, will be able to implement appropriate functionality without undue experimentation.

[0010] References in the specification to "one embodiment", "an embodiment", "an example embodiment", etc., indicate that the embodiment described may 5 include a particular feature, structure, or characteristic, but every embodiment may not necessarily include the particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same embodiment. Further, when a particular feature, structure, or characteristic is 10 described in connection with an embodiment, it is submitted that it is within the knowledge of one skilled in the art to effect such feature, structure, or characteristic in connection with other embodiments whether or not explicitly described.

[0011] Embodiments of the invention may be implemented in hardware, firmware, 15 software, or any combination thereof. Embodiments of the invention may also be implemented as instructions stored on a machine-readable medium, that may be read and executed by one or more processors. A machine-readable medium may include any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computing device). For example, a 20 machine-readable medium may include read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other forms of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.) and others.

[0012] An embodiment of a virtual machine cluster environment 1 is shown in Fig.

1. The virtual machine cluster environment 1 may comprise a plurality of client platforms 10/30 and a server platform 20 that may connect with the client platforms 10/30 through a network 40, such as Ethernet, fiber channel or other like communications link. In the embodiment, the virtual machine cluster environment 1 may comprise any number of client platforms and server platforms.

[0013] Each of the client platforms 10/30 may host a plurality of virtual machines running their own guest operating system(s) and guest application software.

10 The server platform 20 may provide a variety of services to the virtual machines of the client platforms, for example, but not limited to, device virtualization/simulation services and virtual machine management services. A non-exhaustive list of examples for the client platform 10/30 may include mainframe computer, mini-computer, personal computer, portable computer, 15 laptop computer and other devices for transceiving and processing data. A non-exhaustive list of examples for the server platform 20 may include supercomputer, mainframe computer, workstation and other devices for transceiving and processing data.

[0014] An embodiment of structures for the client platforms 10/30 and the server 20 in the virtual machine cluster environment 1 is shown in Fig. 2. The client platforms 10/30 and server platform 20 may be configured according to the hybrid virtual machine monitor architecture.

[0015] In the embodiment, each of the client platforms 10/30 may comprise an underlying hardware machine 11/31 having one or more processors, memory, 25 I/O devices, etc., wherein the I/O devices may further comprise a network

interface to transceive data through the network 40, such as the network interface 110 of the client platform 10 and the network interface 310 of the client platform 30.

[0016] Each of the client platforms 10/30 may further comprise a hypervisor (or 5 micro-hypervisor) 12/32, which is a kernel component configured according to the hybrid virtual machine monitor architecture and responsible for processor(s)/memory virtualization, interrupt handling and domain scheduling. The hypervisor 12/32 may further comprise a network interface management core 120/320 to manage the network interface 110/310 of the underlying 10 machine. Examples for the hypervisor may comprise Xen's hypervisor, Virtual PC's hypervisor, and the like.

[0017] Each of the client platforms 10/30 may further comprise a plurality of guest 15 software images running on a plurality of virtual machines created and managed by the hypervisor 12/32, for example, guest operating systems 131/141 and guest applications 132/142 running on the virtual machines 13/14 of the client platform 10, and guest operating systems 331/341 and guest applications 332/342 running on the virtual machines 33/34 of the client 20 platform 30. The guest operating systems 131, 141, 331 and 341 may be different with each other. The number of guest operating systems running on the client platform may depend on utilization of processor/memory resources 25 and device resources. For example, the number of guest operating systems may be low if the client platform utilizes the processor/memory resources more intensively compared to device utilization. However, the number of guest operating systems may be high if the client platform utilizes the device resources more intensively compared to processor/memory utilization.

Examples for the guest operating system may comprise Linux, Windows, FreeBSD Unix, etc.

[0018] The server platform 20 may comprise an underlying hardware machine 21 having one or more processors, memory, I/O devices, etc., wherein the I/O devices 5 may further comprise a network interface 210 to transceive data through the network 40. The server platform 20 may be loaded with a service operating system 22 configured according to the hybrid virtual machine monitor architecture. The service operating system 22 may respectively interact with the hypervisors 12/32 of the client platforms 10/30 via the network 10 40 to service the plurality of virtual machines in the client platforms 10/30.

Examples for the service operating system may comprise Linux, Windows, FreeBSD Unix, etc.

[0019] The server platform 20 may be further loaded with a plurality of device models 23 and a control panel 24. The device models 23 may be virtual devices 15 that may be created or defined for example according to the hybrid virtual machine monitor architecture. In some embodiments, because of device virtualization/simulation nature, the device models 23 may not be connected to or represented by a real instance of a device, and may not be reflected in a real device that is connected to a hardware component. Examples for the 20 device models 23 may comprise, but not limited to, various I/O devices, an interrupt controller, an event timer, etc.

[0020] The device models 23 may work with the service operating system 22 to provide device virtualization/simulation service to the virtual machines of the client platforms 10/30 via the network 40, including hosting device related 25 operation from/to the virtual machines, such as device related I/O operation

and device model initiated interrupt. In some embodiments, the device models in the virtual machine cluster environment may be centralized in the server platform 20. However, in other embodiments, the client platform 10/30 may keep some of the device models, such as the device models frequently used by the virtual machines of the client platform. For example, the client platform 10/30 may keep the device models inside of the hypervisor 11/31.

[0021] The control panel 24 may work with the service operating system 22 to manage the virtual machines of the client platforms 10/30 via the network 40. In an embodiment, the control panel 24 may comprise a database 241 containing identifiers of the virtual machines (or the guest operating systems running on the virtual machines), the hypervisors and the client platforms in the virtual machine cluster environment that are served by the server platform 20; and as another possibility the database can be created or updated on fly. In an embodiment, a virtual machine of the client platform 10/30 may send a request for a device related input/output operation to the server platform 20 via the network 40. The control panel 24 and the service operating system 22 may identify a corresponding device model from the plurality of device models 23 to host the request, and further construct a feedback packet by incorporating a result of the input/output operation from the device model with identifier(s) to identify the virtual machine that has sent the operation request. In another embodiment, a device model may initiate an interrupt for a virtual machine of the client platform 10/30. The control panel 24 and the service operating system 22 may construct an interrupt packet by incorporating the interrupt instruction with identifier(s) to identify the virtual machine that may handle the interrupt. The identifier(s) may comprise a virtual machine identifier, a

hypervisor identifier and/or client platform identifier. In addition, the control panel 24 may perform virtual machine management for other aspect, such as virtual machine operation management including virtual machine launching management, virtual machine shutdown management, virtual machine migration management, virtual machine suspend management, etc.

5 [0022] The server platform 20 may further be loaded with a plurality of application software running on the service operating system 22, such as applications 25 and 26.

10 [0023] Other embodiments may implement other technologies to the structure depicted in Fig. 2. In an embodiment, the server platform 20 may further comprise a hypervisor running between the underlying hardware machine 21 and the service operating system 22. In such case, the server platform 20 may further comprise one or more virtual machine(s) defined by the hypervisor and the service operating system. In another embodiment, console devices (e.g., 15 frequently used input/output devices such as a keyboard, a mouse, a video device, etc.) may be centralized in the underlying hardware machine 21 of the server platform 20. In such case, the client platform 10/30 may not be provided with the console devices.

20 [0024] Fig. 3 shows an embodiment of a method of implementing a device related input/output operation in the virtual machine cluster environment 1.

25 [0025] In the embodiment, an unauthorized input/output operation for inputting a data from a device (input operation) or outputting a data to the device (output operation) happens in a guest operating system running on a virtual machine of a client platform (e.g., guest operating system 131 running on the virtual machine 13 of the client platform 10), and a corresponding device driver in the

guest operating system may execute an 'IN' instruction for input operation or 'OUT' instruction for output operation, which may cause a trap into the hypervisor of the client platform in block 301, because the guest operating system is depriviledged.

5[0026] In block 302, the hypervisor of the client platform may perceive the unauthorized I/O operation happened in the guest operating system through a virtual machine exit, which is a transition from non-root VMX operation in the virtual machine to root VMX operation in the hypervisor.

[0027] In block 303, the hypervisor of the client platform may construct an I/O request packet and send the packet to the server platform 20 through the network 40. In an embodiment, the I/O request may further comprise identifier(s) to identify a device model in the server platform 20 corresponding to the device that is related to the I/O operation in the guest operating system of the virtual machine. The identifier(s) may comprise a device model identifier and/or a server platform identifier (e.g., a network ID for the server platform). In response to receiving the I/O request packet, the service operating system 22 may identify the corresponding device model from the plurality of device models 23 under control from the control panel 24 and route the I/O request packet to the identified device model in block 304.

20[0028] In block 305, the identified device model may handle the I/O request. For output request, the data in the I/O request packet may be output to the identified device model. Depending upon inter-platform communication interface definition, the server platform 20 may send back a finish notification to the I/O requester (i.e., the client platform 10/30) to notify finish of the 'OUT' operation. However, for input request, the identified device model 23 may

obtain a feedback to the input request, for example, through cooperating with the service operating system 22 and the underlying hardware 21 of the service platform 20, and send a feedback packet to the service operating system 22 in block 306. The feedback packet may incorporate the feedback with identifier(s) to identify the guest operating system (or virtual machine) of the client platform that is executing the I/O operation. In an embodiment, the identifier(s) may comprise a guest operating system identifier (or virtual machine identifier) and/or a client platform identifier (e.g., a network ID for the client platform). In another embodiment, if the client platforms in the virtual machine cluster environment 1 comprises different types of hypervisors, e.g., Xen hypervisor, Virtual PC's hypervisor, etc., then the identifier may further comprise a hypervisor identifier to identify the different hypervisors.

[0029] In block 307, under control from the control panel stored with identifiers of the guest operating systems (or virtual machines), the hypervisors and the client platforms in the virtual machine cluster environment 1, the service operating system 22 may route the feedback packet to the client platform that has sent the I/O request based upon the identifier(s) in the feedback packet. In block 308, the hypervisor of the client platform may identify the guest operating system (or virtual machine) executing the corresponding I/O operation and send the feedback packet to the identified guest operating system through a virtual machine entry, which is another transition from the root VMX operation in the hypervisor to the non-root VMX operation in the virtual machine.

[0030] An embodiment of a method of implementing a device model initiated interrupt in the virtual machine cluster environment 1 is shown in Fig. 4.

[0031] In block 401, one of the device models 23 in the server platform 20 initiate an interrupt for a guest operating system of a client platform in the virtual machine cluster environment 1. In block 402, the device model passes an interrupt packet incorporating the interrupt instruction as well as identifier(s) to 5 identify the guest operating system (or virtual machine) of the client platform to the service operating system 22. In an embodiment, the identifier(s) may comprise a guest operating system identifier (or virtual machine identifier) and/or a client platform identifier (e.g., a network ID for the client platform). In another embodiment, if the client platforms in the virtual machine cluster 10 environment 1 comprises different types of hypervisors, e.g., Xen hypervisor, Virtual PC's hypervisor, etc., then the identifier may further comprise a hypervisor identifier to identify the different hypervisors.

[0032] In block 403, under control from the control panel 24, the service operating system 22 may send the interrupt packet to the client platform identified by the 15 identifier(s) in the interrupt packet. The hypervisor of the client platform may further identify the guest operating system based upon the interrupt packet and inject the interrupt to the identified guest operating system through the virtual machine entry (block 404), so that the guest operating system may handle the interrupt (block 405).

20[0033] Fig. 5 shows an embodiment of a computer platform that may be implemented as a platform in the virtual machine cluster environment 1.

[0034] In an embodiment, the computing platform may comprise one or more processors 50, memory 51, chipset 52, I/O device 53, BIOS firmware 54 and the like. The one or more processors 50 are communicatively coupled to 25 various components (e.g., the memory 51) via one or more buses such as a

processor bus. The processors 50 may be implemented as an integrated circuit (IC) with one or more processing cores that may execute codes under a suitable architecture, for example, including Intel® Xeon™, Intel® Pentium™, Intel® Itanium™ architectures, available from Intel Corporation of Santa Clara, California.

5

[0035] In an embodiment, the memory 51 may store codes to be executed by the processor 50. A non-exhaustive list of examples for the memory 51 may comprise one or a combination of the following semiconductor devices, such as synchronous dynamic random access memory (SDRAM) devices, 10 RAMBUS dynamic random access memory (RDRAM) devices, double data rate (DDR) memory devices, static random access memory (SRAM), flash memory devices, and the like.

[0036] In an embodiment, the chipset 52 may provide one or more communicative path among the processor 50, memory 51 and various components, such as 15 the I/O device 53, peripheral component 54, and BIOS firmware 55. The chipset 52 may comprise a memory controller hub 520, an input/output controller hub 521 and a firmware hub 522.

[0037] In an embodiment, the memory controller hub 520 may provide a 20 communication link to the processor bus that may connect with the processor 50 and to a suitable device such as the memory 51. The memory controller hub 520 may couple with the I/O controller hub 521 that may provide an interface to the I/O devices 53. A non-exhaustive list of examples for the I/O devices 53 may comprise a keyboard, a mouse, a storage device, a video device, an audio device, a network interface, and the like.

[0038] In an embodiment, the memory controller hub 520 may communicatively couple with a firmware hub 522 via the input/output controller hub 521. The firmware hub 522 may couple with the BIOS firmware 54 that may store routines that the computing platform executes during system startup in order to initialize the processors 50, chipset 52, and other components of the computing platform. Moreover, the BIOS firmware 54 may comprise routines or drivers that the computing platform may execute to communicate with one or more components of the compute platform.

[0039] In an embodiment, the computer platform as depicted in Fig. 5 may perform as a client 10/30. The memory 51 may store software images such as a hypervisor, guest operating systems, guest applications, etc. In another embodiment, the computer platform as depicted in Fig. 5 may perform as a server platform 20. The memory 51 may store software images such as a service operating system, device models, control panel, and optionally various applications, etc.

[0040] Another embodiment for the server platform and client platform structures in the virtual machine cluster environment 1 is depicted in Fig. 6. As depicted, more than one server platforms (e.g., owner server platform 620 and I/O server platform 6200) may connect with the client platform 10/30 through the network 40 to provide them with device virtualization/simulation services. The owner server platform 620 has similar structure as the server platform 20 and the I/O server platform 6200 is loaded with a plurality of device models 6230 so that the server platforms 620/6200 may provide robust device virtualization services to their client platforms. Functionalities of the control panel 624 may encompass management of the device models 6230 in the I/O server platform

6200. In an embodiment, the control panel 624 may perform the device model identification based upon an I/O request packet from the client platform 10/30. In such case, the owner server platform 620 may first receive the I/O request packet from the client platform 10/30 via the network 40 and route the I/O request packet to the device model 6230 in the I/O service platform 6200 after 5 analyzing the identifier(s) in the I/O request packet. In another embodiment, the device model 6230 may construct an I/O feedback packet or an interrupt packet by incorporating an identifier(s) to identifier the guest operating system (or virtual machine) in the intended client platform 10/30 under supervision 10 from the control panel 624.

[0041] While certain features of the invention have been described with reference to example embodiments, the description is not intended to be construed in a limiting sense. Various modifications of the example embodiments, as well as other embodiments of the invention, which are apparent to persons skilled in 15 the art to which the invention pertains are deemed to lie within the spirit and scope of the invention. For example, although the above embodiments are described according to the hybrid virtual machine monitor architecture, the present invention may be applied to other kinds of virtual machine monitor architectures, such as a host virtual machine monitor.

What is claimed is:

1. A method for a client platform coupled to a server platform via a network, comprising:
 - 5 determining that an input/output operation related to a hardware device happens in a virtual machine of the client platform; and requesting the server platform via the network to handle the input/output operation related to the hardware device.
 - 10 2. The method of claim 1, wherein the request comprises a server identifier to identify the server platform.
 - 15 3. The method of claim 1, wherein the request comprises a software device identifier to identify a software device from a plurality of software devices in the server platform to handle the input/output operation related to the hardware device.
 - 20 4. The method of claim 1, further comprising: receiving a feedback for the input operation from the server through the network, the feedback comprising a virtual machine identifier to identify the virtual machine in the client platform that is executing the input operation; and sending the feedback to the virtual machine identified by the virtual machine identifier.

5. The method of claim 1, further comprising:
 - receiving via the network an interrupt instruction issued by a software device of the server platform, the interrupt instruction comprising a virtual machine identifier to identify a virtual machine to perform the interrupt instruction; and
 - 5 injecting the interrupt instruction into the virtual machine identified by the virtual machine identifier.
6. A machine-readable medium comprising a plurality of instructions which when executed result in a client platform:
 - 10 determining that an input/output operation related to a hardware device happens in a virtual machine of the client platform; and
 - requesting a server platform coupled to the client platform via the network to handle the input/output operation related to the hardware device.
- 15 7. The machine-readable medium of claim 6, wherein the request further comprise a server identifier to identify the server platform.
8. The machine-readable medium of claim 6, wherein the request comprises a software device identifier to identify a software device from a plurality 20 of software devices in the server platform to handle the input/output operation related to the hardware device.

9. The machine-readable medium of claim 6, wherein the plurality of instructions further result in the client platform:

receiving a feedback for the input operation from the server platform through the network, the feedback comprising a virtual machine identifier to

5 identify the virtual machine in the client platform that is executing the input operation; and

sending the feedback to the virtual machine identified by the virtual machine identifier.

10 10. The machine-readable medium of claim 6, wherein the plurality of instructions further result in the client platform:

receiving an interrupt instruction issued by a software device from the plurality of software devices in the server platform through the network, the interrupt instruction comprising a virtual machine identifier to identify a virtual

15 machine to perform the interrupt instruction;

injecting the interrupt instruction to the virtual machine identified by the virtual machine identifier.

11. A method for a server platform coupled to a client platform via a network,

20 receiving, from the client platform via the network, a request for an input/output operation related to a hardware device; and

identifying a software device from a plurality of software devices in the server platform to handle the request, the identified software device corresponding to the hardware device related to the input/output operation.

12. The method of claim 11, wherein the request comprises a software device identifier to identify the software device in the server platform.

13. The method of claim 11, further comprising
5 receiving the request for the input/output operation with a first server of the server platform,

determining whether the identified software device is in a second server of the server platform; and

10 sending the request from the first server to the second server via the network, in response to determining that the identified software device is in the second server.

14. The method of claim 11, further comprising:

obtaining a result for the input operation from the identified software device;
15 constructing a feedback with the result and a virtual machine identifier to identify a virtual machine in the client platform that is executing the input operation; and

sending the feedback from the server platform to the client platform through the network.

20 15. The method of claim 14, wherein the feedback further comprise a client platform identifier to identify the client platform that has sent the request.

16. The method of claim 11, further comprising:

25 issuing an interrupt instruction from a software device of the plurality of software devices in the server platform to the client platform through the network.

17. The method of claim 11, wherein the interrupt instruction further comprises a virtual machine identifier to identify a virtual machine in the client platform to handle the interrupt.

5 18. A machine-readable medium comprising a plurality of instructions which when executed result in a server platform:

receiving a request for an input/output operation related to a hardware device from a client platform through a network; and

10 identifying a software device from a plurality of software devices in the server platform to handle the request, the identified software device corresponding to the hardware device related to the input/output operation.

15 19. The machine readable medium of claim 18, wherein the request comprises a software device identifier to identify the software device in the server platform.

20 20. The machine readable medium of claim 18, wherein the plurality of instructions that result in the server platform receiving the request, further result in the server platform:

20 receiving the request for the input/output operation from the client platform with a first server in the server platform.

21. The machine readable medium of claim 20, wherein the plurality of instructions further result in the server platform:

determining whether the identified software device is in a second server of the server platform; and

5 sending the request from the first server to the second server through the network, in response to determining that the identified software device is in the second server.

22. The machine readable medium of claim 18, wherein the plurality of instructions further result in the server platform:

10 obtaining a result for the input operation from the identified software device; constructing a feedback with the result and a virtual machine identifier to identify a virtual machine in the client platform that is executing the input operation; and

15 sending the feedback from the server platform to the client platform through the network.

23. The machine readable medium of claim 22, wherein the feedback further comprise a client identifier to identify the client platform that has sent the request.

24. The machine readable medium of claim 18, wherein the plurality of instructions further result in the server platform:

issuing an interrupt instruction from a software device of the plurality of 25 software devices in the server platform to the client platform through the network.

25. The machine readable medium of claim 24, wherein the interrupt instruction further comprises a virtual machine identifier to identify a virtual machine in the client platform to handle the interrupt.

5 26. A system, comprising

 a client platform comprising:

 a plurality of virtual machines; and

 a virtual machine monitor to determine that an input/output operation related to a hardware device happens in a virtual machine of the plurality of virtual

10 machines and construct a request for the input/output operation;

 a client network interface to send the request through a network; and

 the server platform comprising:

 a server network interface to receive the request through the network;

 a plurality of software devices;

15 a controller to identify a software device from the plurality of software devices to handle the request, the identified software device corresponding to the hardware device related to the input/output operation.

27. The system of claim 26, wherein the request further comprises a

20 software identifier to identify the software device in the server platform.

28. The system of claim 26, wherein
the identified software device in the server platform is further to obtain a
result for the input operation, and construct a feedback with the result and a virtual
machine identifier to identify the virtual machine in the client platform under control
5 from the controller, and
the server network interface is further to send the feedback to the client
platform through the network.

29. The system of claim 26, wherein
10 the client network interface is further to receive a feedback for the input
operation from the server platform through the network; and
the virtual machine monitor is further to identify the virtual machine in the
client platform that is executing the input operation based upon the feedback and
send the feedback to the identified virtual machine.

15 30. The system of claim 26, wherein
a software device in the server platform is to issue an interrupt instruction
under control from the controller, the interrupt instruction including a virtual
machine identifier to identify another virtual machine in the client platform to
20 handle the interrupt instruction; and
the server network interface is further to send the interrupt instruction to the
client platform through the network.

31. The system of claim 30, wherein
the client network interface is further to receive the interrupt instruction; and
the virtual machine monitor is further to identify the another virtual machine
from the plurality of virtual machines based upon the interrupt instruction and
5 inject the interrupt into the identified another virtual machine.

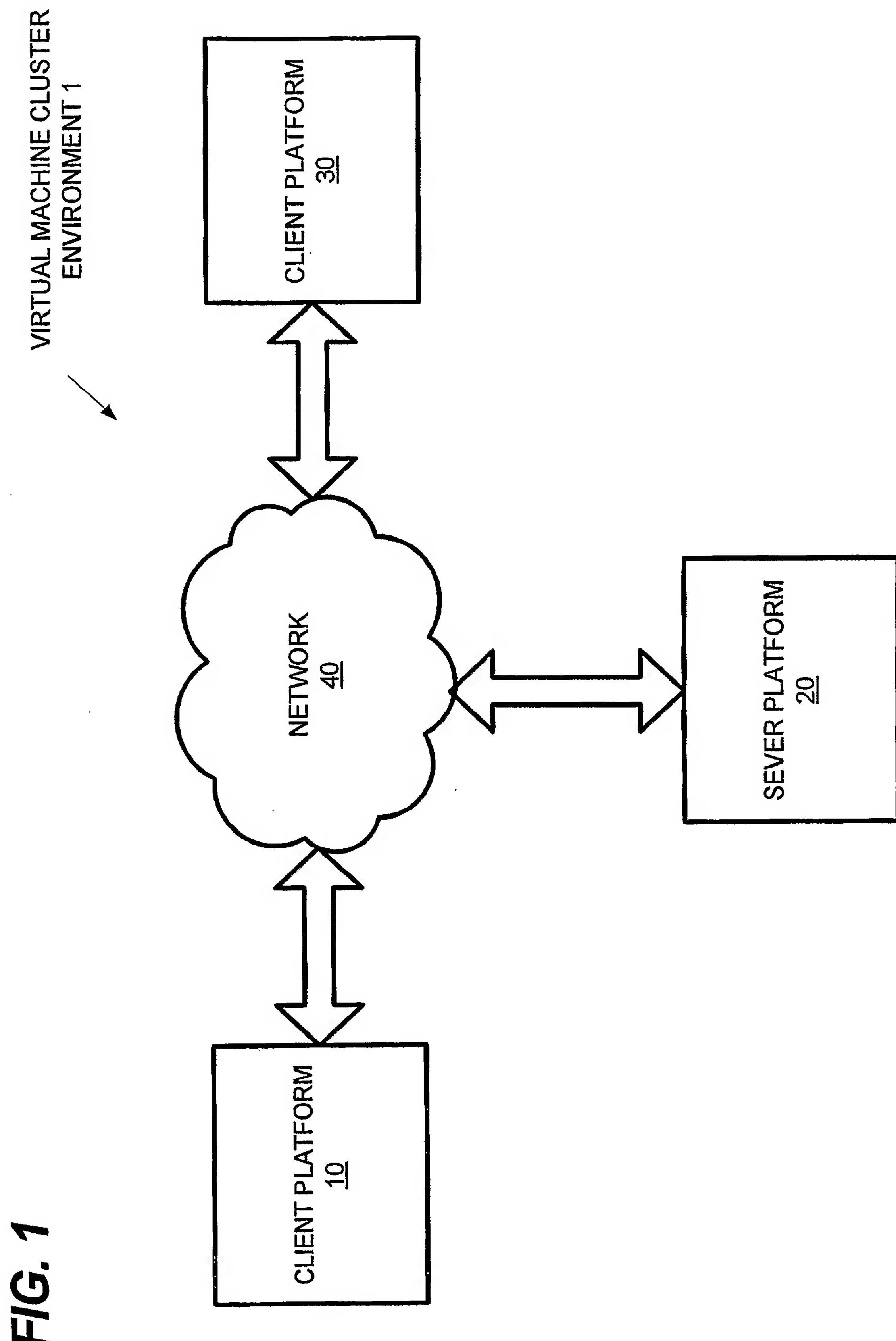
32. A method for a system comprising a client platform and a server
platform, wherein the client platform couples to the server platform through a
network, the method comprising:
10 determining that an input/output operation related to a hardware device
happens in a virtual machine of the client platform;
sending a request for the input/output operation from the client platform to
the server platform through the network;
receiving the request through the network by the server platform;
15 identifying a software device from a plurality of software devices in the
server platform to handle the request, wherein the identified virtual device
corresponds to the hardware device related to the input/output operation.

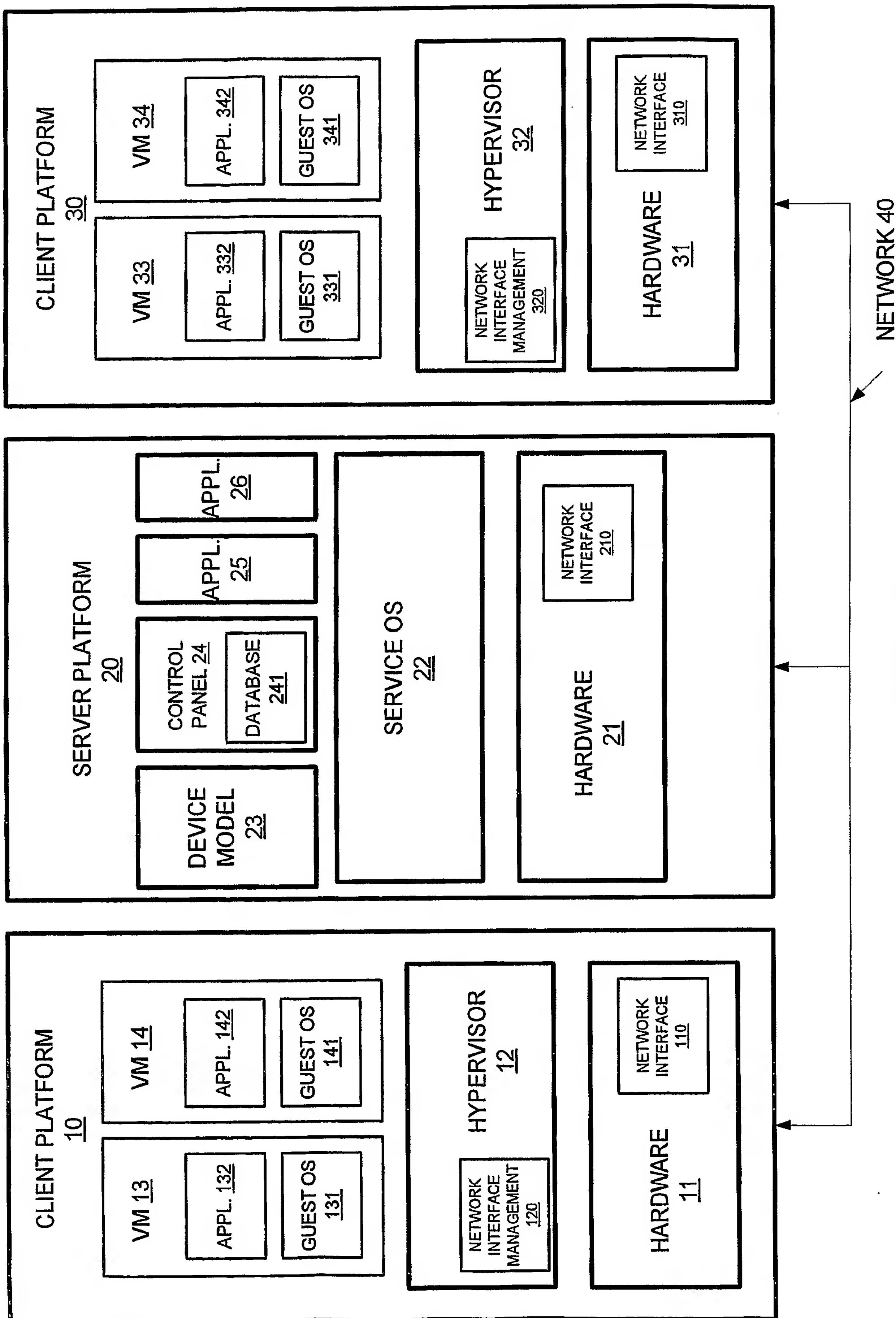
33. The method of claim 32, wherein the receiving further comprises
20 receiving the request with a first server in the server platform.

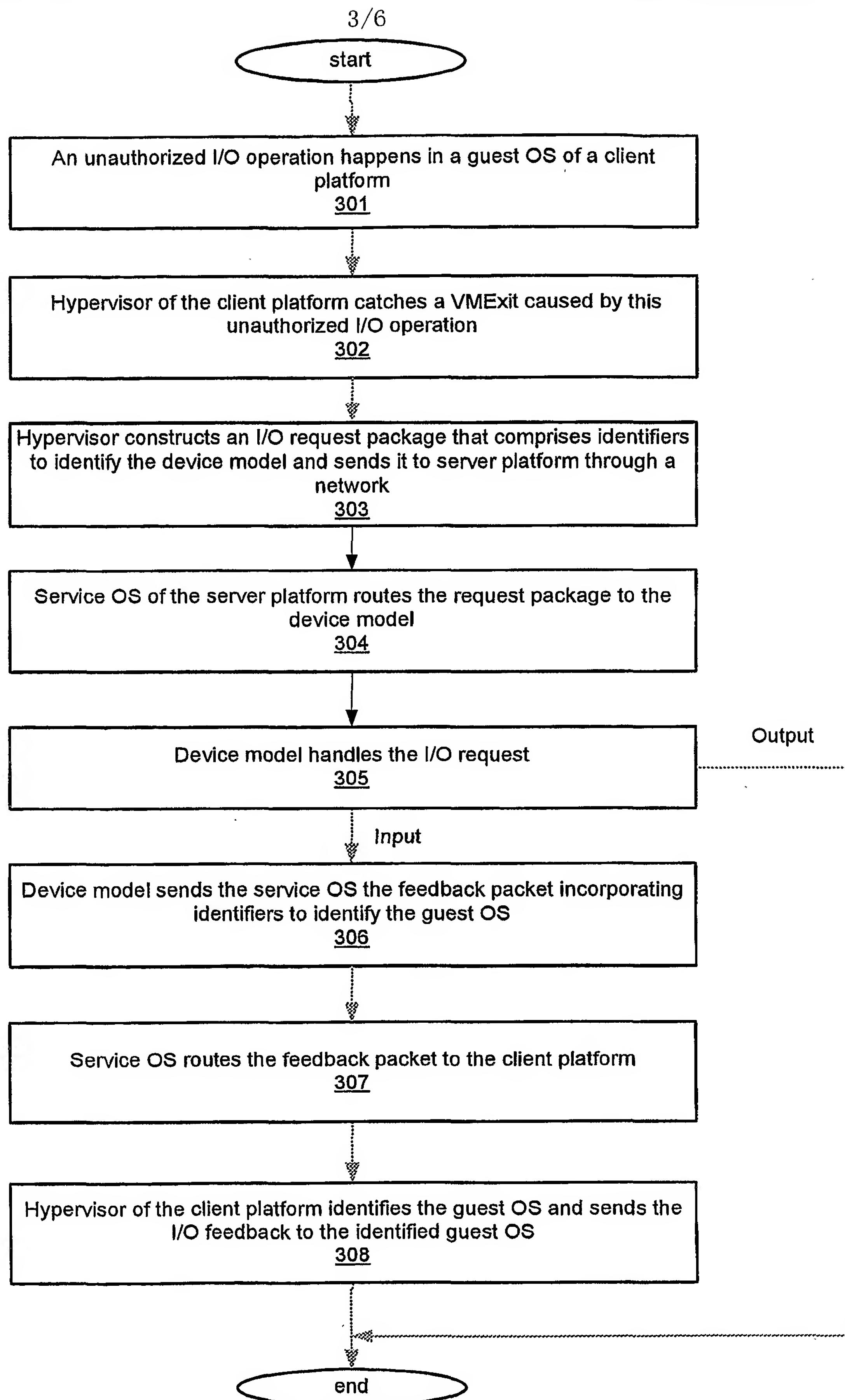
34. The method of claim 33, further comprising:
 - determining whether the identified software device is in a second server of the server platform; and
 - sending the request from the first server to the second server through the network, in response to determining that the identified software device is in the second server.
35. The method of claim 32, further comprising:
 - obtaining a result for the input operation from the identified software device in the server platform;
 - constructing a feedback with the result and a virtual machine identifier to identify the virtual machine in the client platform that is executing the input operation; and
 - sending the feedback from the server platform to the client platform through the network.
36. The method of claim 35, wherein the feedback further comprise a client identifier to identify the client platform that has sent the request.
37. The method of claim 32, further comprising:
 - issuing an interrupt instruction from a software device in the server platform to the client platform through the network.
38. The method of claim 32, wherein the interrupt instruction further comprises a virtual machine identifier to identify another virtual machine in the client platform to handle the interrupt.

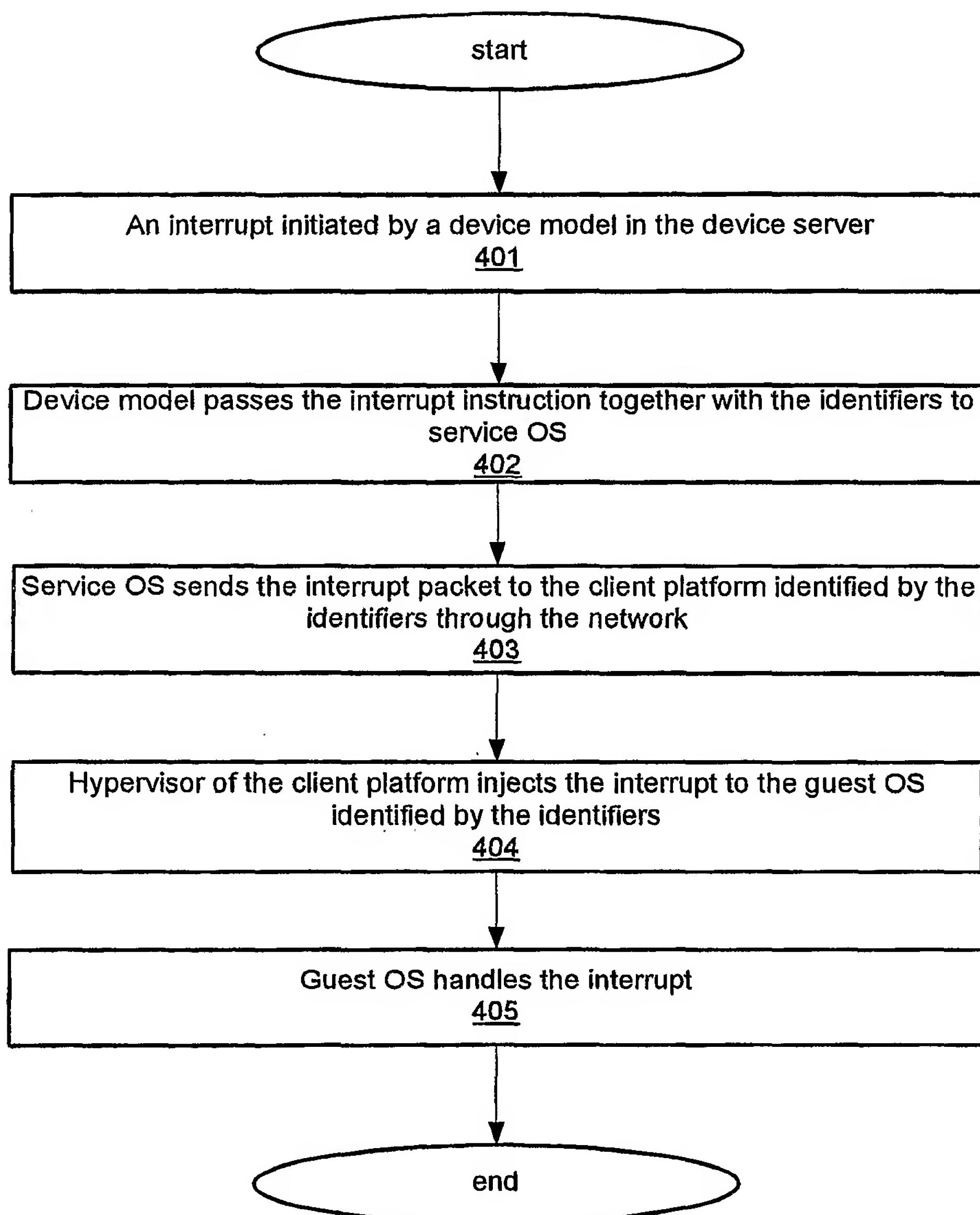
39. The method of claim 38, further comprising:
 - receiving the interrupt instruction through the network by the client platform;
 - identifying the another virtual machine in the client platform based upon the interrupt instruction; and

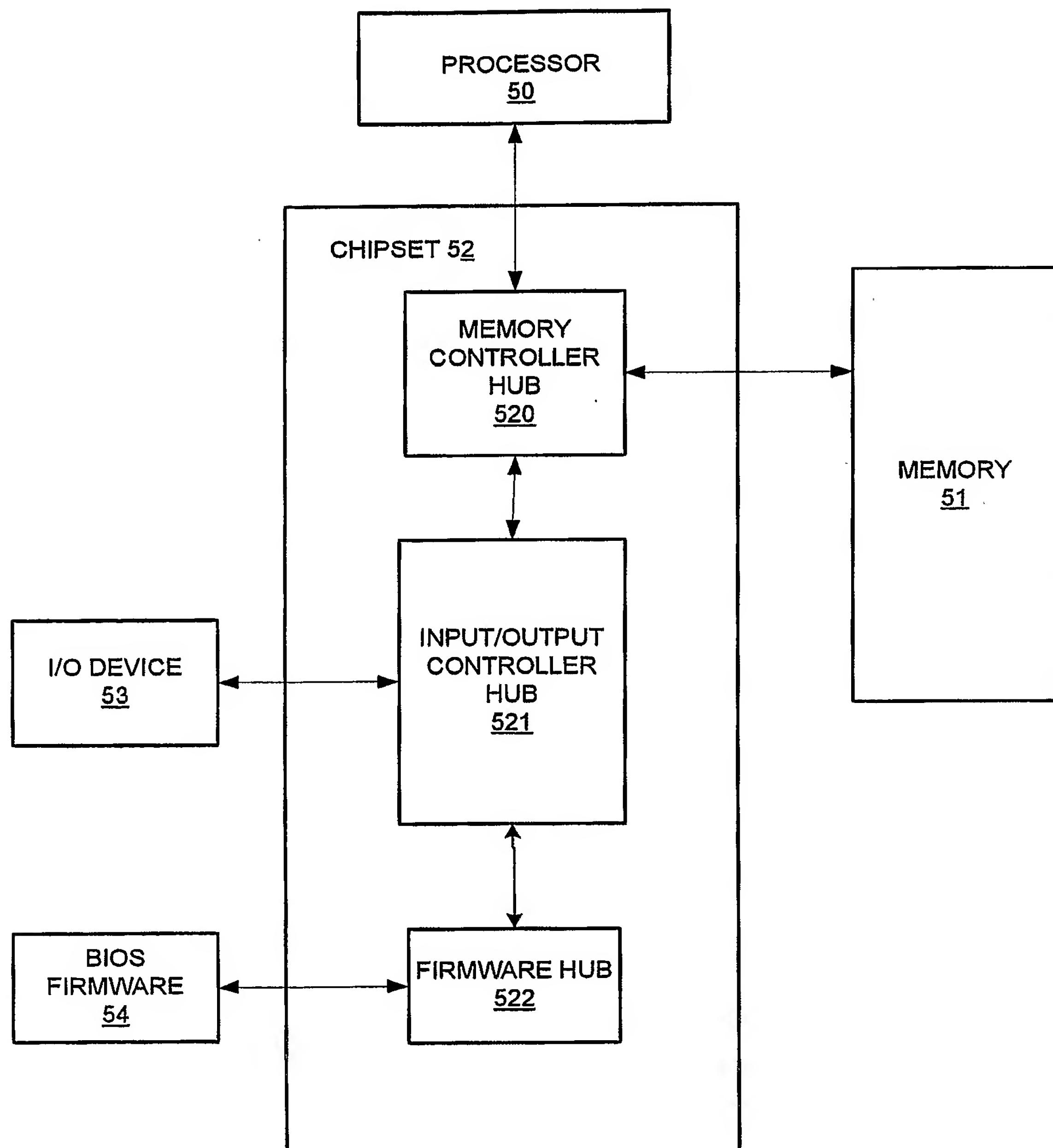
5 injecting the interrupt into the identified another virtual machine.

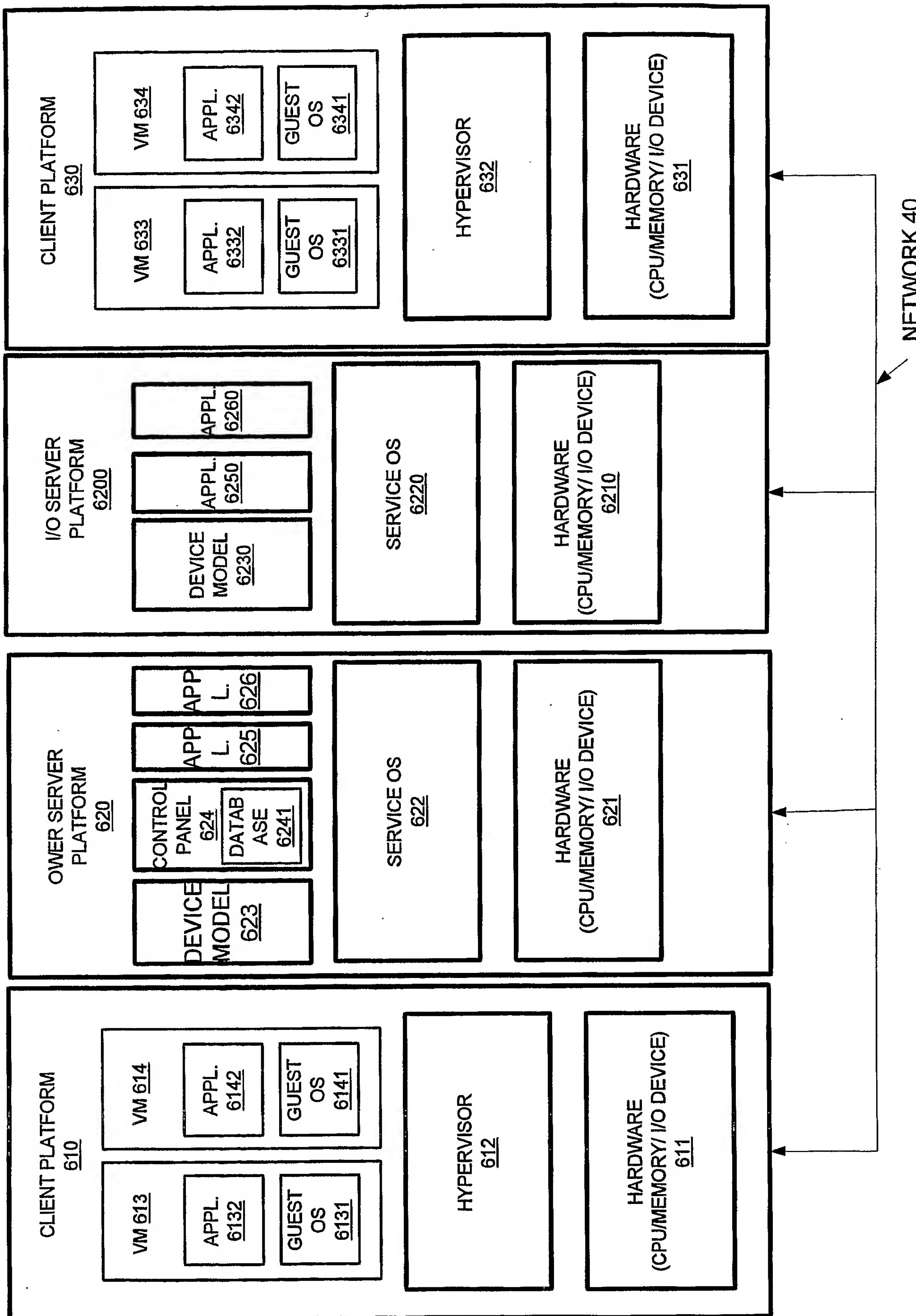
**FIG. 1**



**FIG. 3**

**FIG. 4**

**FIG. 5**

**FIG. 6**

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2005/002304

A. CLASSIFICATION OF SUBJECT MATTER

G06F15/16(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 8 G06F15/, G06F9/, G06F12/

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) WPI
EPODOC PATENTPIC PAJ CNPAT device, model, virtual, machine, cluster, client, server, input, output, hardware, network, platform

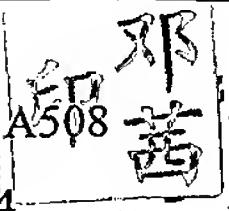
C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN1012131A, HITACHI LTD, 15.Jul 1987(15.07.1987), the whole document	1-39
A	JP58195254A, NIPPON ELECTRIC CO, 14.Nov 1983(14.11.1983), the whole document	1-39
A	US6799195B1, INVENSYS SYSTEMS INC, 28.Sep 2004(28.09.2004), the whole document	1-39

Further documents are listed in the continuation of Box C.

See patent family annex.

* Special categories of cited documents:	
"A" document defining the general state of the art which is not considered to be of particular relevance	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"E" earlier application or patent but published on or after the international filing date	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L" document which may throw doubts on priority claim (S) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O" document referring to an oral disclosure, use, exhibition or other means	"&" document member of the same patent family
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search 18.Sep 2006(18.09.2006)	Date of mailing of the international search report 26 · OCT 2006 (26 · 10 · 2006)
Name and mailing address of the ISA/CN The State Intellectual Property Office, the P.R.China 6 Xitucheng Rd., Jimen Bridge, Haidian District, Beijing, China 100088 Facsimile No. 86-10-62019451	Authorized officer Telephone No. 86-10-62084944 

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2005/002304

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
CN1012131A	15.Jul 1987(15.07.1987)	EP0210640 A US4887202 A EP0210640 B1 KR9308265 B1 JP62031437A JP7092761B B JP2057991C C DE3687805D D DE3687805T T	04. Feb 1987(04. 02.1987) 12.Dec 1989 (12.12.1989) 24.Feb 1993 (24.02.1993) 27.Aug 1993(27.08.1993) 10.Feb 1987(10.02.1987) 09.Oct 1995(09.10.1995) 10. Jun 1996(10. 06.1996) 01.Apr 1993 (01.04.1993) 09.Jun 1993(09.06.1993)
JP58195254A	14.Nov 1983(14.11.1983)	none	
US6799195B1	28.Sep 2004(28.09.2004)	US2004230643 A1 EP0825506 A2	18.Nov 2004 (18.11.2004) 25.Feb 1998 (25.02.1998)